An Epistemic Characterization of Zero Knowledge

Joseph Y. Halpern, Rafael Pass, and Vasumathi Raman
Computer Science Department
Cornell University
Ithaca, NY, 14853, U.S.A.
e-mail: {halpern, rafael, vraman}@cs.cornell.edu

March 16, 2009

Abstract

Halpern, Moses and Tuttle presented a definition of interactive proofs using a notion they called *practical knowledge*, but left open the question of finding an epistemic formula that completely characterizes zero knowledge; that is, a formula that holds iff a proof is zero knowledge. We present such a formula, and show that it does characterize zero knowledge. Moreover, we show that variants of the formula characterize variants of zero knowledge such as *concurrent zero knowledge* [Dwork, Naor, and Sahai 1998] and *proofs of knowledge* [Feige, Fiat, and Shamir 1987; Tompa and Woll 1987].

1 Introduction

The notions of *interactive proof* and *zero knowledge* were introduced by Goldwasser, Micali, and Rackoff [1989], and have been the subject of extensive research ever since. Informally, an interactive proof is a two-party conversation in which a "prover" tries to convince a polynomial-time "verifier" of the truth of a fact φ (where φ typically has the form $x \in L$, where x is a string and L is a *language* or set of strings) through a sequence interactions. An interactive proof is said to be zero knowledge if, whenever φ holds, the verifier has an algorithm to generate on its own the conversations it could have had with the prover during an interactive proof of φ (according to the correct distribution of possible conversations). Intuitively, the verifier does not learn anything from talking to the prover (other than φ) that it could not have learned on its own by generating the conversations itself. Consequently, the only knowledge gained by the verifier during an interactive proof is that φ is true. The notion of "knowledge" used in zero knowledge is based on having an algorithm to generate the transcript of possible conversations with the prover; the zero-knowledge condition places a restriction on what the verifier is able to generate after interacting with the prover (in terms of what he could generate before). The relationship between this ability to generate and logic-based notions of knowledge is not immediately obvious. Having a logic-based characterization of zero knowledge would enhance our understanding and perhaps allow us to apply model-checking tools to test whether proofs are in fact zero knowledge. However, getting such a characterization is not easy. Since both probability and the computational power of the prover and verifier play crucial roles in the definition of zero knowledge, it is clear that the standard notion of knowledge (truth in all possible worlds) will not suffice.

Halpern, Moses and Tuttle [1988] (HMT from now on) were the first to study the relationship between knowledge and being able to generate. They presented a definition of interactive proofs using a notion they called *practical knowledge*. They proved that, with high probability, the verifier in a zero-knowledge proof of $x \in L$ practically knows a fact ψ at the end of the proof iff it practically knows $x \in L \Rightarrow \psi$ at the beginning of the proof; they call this property *knowledge security*. Intuitively, this captures the idea that zero knowledge proofs do not "leak" knowledge of facts other than those that follow from $x \in L$. They also define a notion of knowing how to generate a y satisfying a relation $R(x, y)$, and prove that, with high probability, if the verifier in a zero-knowledge proof of $x \in L$ knows how to generate a y satisfying $R(x, y)$ at the end of the proof, then he knows how to do so at the beginning as well; they called this property *generation security*. This captures the intuition that at the end of a zero-knowledge proof, the verifier cannot do anything that it could not do at the beginning.

HMT left open the question of finding an epistemic formula that completely characterizes zero knowledge; that is, a formula that holds iff a proof is zero knowledge [Goldwasser, Micali, and Rackoff 1989]. In this paper we present a strengthening of knowledge security and generation security that we call *relation hiding*, which we show does characterize zero knowledge. Moreover, we show that variants of relation hiding characterize variants of zero knowledge such as *concurrent zero knowledge* [Dwork, Naor, and Sahai 1998] and *proofs of knowledge* [Feige, Fiat, and Shamir 1987; Tompa and Woll 1987].

2 Background

In this section, we review the relevant background both in cryptography (interactive proof systems and zero knowledge) and epistemic logic (specifically, modeling knowledge and probability using the runs and systems framework [Fagin, Halpern, Moses, and Vardi 1995; Halpern and Tuttle 1993]). In addition, we introduce some of the notation that will be needed for our new results.

2.1 Interactive Proof Systems

An *interactive protocol* is an ordered pair (P, V) of probabilistic Turing machines. P and V share a read-only input tape; each has a private one-way, read-only random tape; each has a private work tape; and P and V share a pair of one-way communication tapes, one from P to V being write-only for P and read-only for V, and the other from V to P being write-only for V and read-only for P. An *execution* of the protocol (P, V) is defined as follows. At the beginning, the input tape is initialized with some common input x, each random tape is initialized with an infinite sequence of random bits, each work tape may or may not be initialized with an initial string, and the communication tapes are initially blank. The execution then proceeds in a sequence of rounds. During any given round, V first performs some internal computation making use of its work tape and other readable tapes, and then sends a message to P by writing on its write-only communication tape; P then performs a similar computation. Either P or V may halt the interaction at any time by entering a halt state. V accepts or rejects the interaction by entering an accepting or rejecting halt state, respectively, in which case we refer to the resulting execution as either an accepting or rejecting execution. The running time of P and V during an execution of (P, V) is the number of steps taken by P and V respectively, during the execution. We assume that V is a probabilistic Turing machine running in time polynomial in $|x|$, and hence that it can perform only probabilistic, polynomial-time computations during each round. For now we make no assumptions about the running time of P.

Denote by $(P(s) \leftrightarrow V(t))(x)$ the random variable that takes two random strings $\rho_p, \rho_v \in \{0,1\}^*$ as input and outputs an execution of (P, V) in which the prover's work tape is initialized with s, the verifier's work tape is initialized with t, the input tape is initialized with x, and ρ_p, ρ_v are the contents of the prover and verifier's respective random tapes. We can think of s as the prover's auxiliary information, t as the verifier's initial information, and x as the common input. Let $Accepts_v[(P(s) \leftrightarrow V(t))(x)]$ be the random variable that takes two infinite random strings $\rho_p, \rho_v \in \{0,1\}^\infty$ as input, and outputs true iff the verifier enters an accept state at the end of the execution of the protocol (P, V) where ρ_p and ρ_v are the contents of the prover and verifier's respective random tapes, and false otherwise.

Informally, an interactive protocol (P, V) is an interactive proof system for a language L if, when run on input x (and possibly some auxiliary inputs s and t), after the protocol, if the prover and verifier are both "good"—that is, the prover uses P and the verifier uses V—the verifier is almost always convinced that $x \in L$. Moreover, no matter what protocol the prover uses, the verifier will hardly ever be convinced that $x \in L$ if it is not. The "almost always" and "hardly ever" are formalized in terms of negligible functions. A function $\epsilon : \mathbb{N} \rightarrow [0,1]$ is *negligible* if for every positive integer k there exists an $n_0 \in \mathbb{N}$ such that for all $n > n_0$, $\epsilon(n) < \frac{1}{n^k}$; that is, ϵ is eventually smaller than any inverse polynomial. Finally, let $\Pr_{U_k}$ denote the uniform probability over strings in $(\{0,1\}^\infty)^k$. For ease of notation, we typically omit the subscript k when it does not play a significant role or is clear from context, writing just $\Pr_U$.

Definition 1 *An interactive protocol* (P, V) *is an* interactive proof system for language L *if the following conditions are satisfied:*

- *Completeness: There exists a negligible function* ϵ *such that for sufficiently large* $|x|$ *and for every*

s and t, if $x \in L$ then
$\Pr_U[Accepts_v[(P(s) \leftrightarrow V(t))(x)]] \geq 1 - \epsilon(|x|)$.

- Soundness: *There exists a negligible function δ such that for sufficiently large $|x|$, for every protocol P^* for the prover, s, and t, if $x \notin L$ then*
  $\Pr_U[Accepts_v[(P^*(s) \leftrightarrow V(t))(x)]] \leq \delta(|x|)$.

The completeness condition is a guarantee to both the good prover and the good verifier that if $x \in L$, then with overwhelming probability the good prover will be able to convince the good verifier that $x \in L$. The soundness condition is a guarantee to the good verifier that if $x \notin L$, then the probability that an arbitrary (possibly malicious) prover is able to convince the good verifier that $x \in L$ is very low. The probability here is taken over the runs of the protocol where the the verifier's initial information is s, the prover's initial information is t, and x is the common input. The probability is generated by the random coin flips of the prover and verifier (which in turn determine what happens in the run); we do not assume a probability on s, t, or x.

2.2 Zero Knowledge

To make the notion of zero precise, we need a few preliminary definitions. We consider zero-knowledge proofs of languages L that have a *witness relation* R_L, where R_L is a set of pairs (x, y) such that $x \in L$ iff there exists a y such that $(x, y) \in R_L$; let $R_L(x) = \{y : (x, y) \in R_L\}$. Note that all languages in the complexity class $\mathcal{NP}$ have this property. Define $View_v[(P(s) \leftrightarrow V(t))(x)]$ to be the random variable that, on input ρ_p, ρ_v, describes the verifier's *view* in the execution $(P(s) \leftrightarrow V(t))(x)(\rho_1, \rho_2)$, that is, the verifier's initial auxiliary input t, the sequence of messages received and read thus far by the verifier, and the sequence of coin flips used thus far.

The intuition behind zero knowledge is that the reason the verifier learns nothing from an interaction is that he can simulate it. The simulation is carried out by a probabilistic Turing machine. It should be possible to carry out the simulation no matter what algorithm the verifier uses (since we hope to show that, no matter what algorithm the verifier uses, he gains no information beyond the fact that $x \in L$), so we have a simulator S_{V^*} for every algorithm V^* of the verifier. The simulator S_{V^*} actually generates verifier views of the conversations. With *perfect zero knowledge*, the distribution of the views created by S_{V^*} given just inputs x and t (which is all the verifier sees) is identical to the actual distribution of the verifier's views generated by $(P(s) \leftrightarrow V(t))(x)$. With *statistical zero knowledge*, the two distributions are just required to be close. Finally, with *computational zero knowledge*, no PPT (*probabilistic polynomial time*) algorithm can distinguish the distributions. We capture the notion of "distinguishing" here by using a PPT distinguisher D. The distinguisher gets as input verifier views generated by S_{V^*} and by the actual conversation, and must output either 1 or 0, depending on whether it believes the view came from S_{V^*} or the actual conversation. Notice that the inputs to the simulator (x and t) are both accessible by the verifier, so the verifier could, given his initial state and the common input, run the simulator instead of interacting with the prover. The distinguisher tries to identify whether the verifier talked to the prover or ran the simulator on his own. If no distinguisher is able to tell the difference, then the verifier might as well not have interacted with the prover but run the simulator instead; we say that the interaction was "zero-knowledge" in this case because the verifier saw nothing during the interaction that he could not simulate.

We allow the distinguisher to have additional information in the form of auxiliary inputs (in addition to the view it is trying to distinguish). This allows the distinguisher to have information that the verifier never sees, such as information about the prover's state, since such information could be helpful in identifying views from the interaction and telling them apart from those produced by the verifier alone. Allowing the distinguisher to get such auxiliary inputs strengthens the zero knowledge requirement in that, no matter what additional information the distinguisher might have, he cannot tell apart views of the interaction from simulated ones.

Definition 2 *An interactive proof system (P, V) for L is said to be* computational zero knowledge *if, for every PPT verifier protocol V^*, there is a probabilistic Turing machine S_{V^*} that takes as input the common input x and verifier's auxiliary information t, runs in expected time polynomial in $|x|$, and outputs a view for the verifier such that for every PPT (probabilistic polynomial time) Turing machine D that takes as input a view of the verifier and an auxiliary input $z \in \{0,1\}^*$, there exists a negligible function ϵ such that for all $x \in L, s \in R_L(x), t \in \{0,1\}^*, z \in \{0,1\}^*$,*

$$\begin{aligned}&|\Pr_U[D(S_{V^*}(x,t), z) = 1] - \\ &-\Pr_U[D(View_v[(P(s) \leftrightarrow V^*(t))(x)], z) = 1]| \leq \epsilon(|x|).\end{aligned}$$

2.3 The Runs and Systems Framework

Our analysis of interactive proof systems is carried out in *runs and systems* framework [Fagin, Halpern, Moses, and Vardi 1995]. The systems we consider consist of a (possibly infinite) set of communicating agents. Agents share a global clock that starts at time 0 and proceeds in discrete increments of one. Computation in the system proceeds in rounds, round m lasting from time $m-1$ to time m. During a round, each agent first performs some (possibly probabilistic) local computation, then sends messages to other agents, and then receives all messages sent to him during that round. Each agents starts in some initial *local state*; its local state then changes over time. The agent's *local state* at time $m \geq 0$ consists of the time on the global clock, the agent's initial information (if any), the history of messages the agent has received from other agents and read, and the history of coin flips used. A *global state* is a tuple of local states, one for each agent and one for the nature, which keeps track of information about the system not known to any of the agents. We think of each agent as following a protocol that specifies what the agent should do in every local state. An infinite execution of such a protocol (an infinite sequence of global states) is called a *run*. We define a *system* to be a set of such runs, often the set of all possible runs of a particular protocol. Given a run r and a time m, we refer to (r, m) as a *point*, and we say that (r, m) is a point of the system $\mathcal{R}$ if $r \in \mathcal{R}$. We denote the global state at the point (r, m) (that is, the global state at time m in r) by $r(m)$, and the local state of agent a in $r(m)$ by $r_a(m)$. Let $\mathcal{K}_a(r, m) = \{(r', m') : r_a(m) = r'_a(m')\}$; $\mathcal{K}_a(r, m)$ can be thought of as the set of points that a considers possible at (r, m), because he has the same local state at all of them. Since the agent's local state at time m consists of the time on the global clock, any point that a considers possible at (r, m) is also at time m, so $\mathcal{K}_a(r, m) = \{(r', m) : r_a(m) = r'_a(m)\}$.

In interactive proof systems, we assume that there are two agents—a prover p and a verifier v. Both agents have a common input (typically a string $x \in \{0,1\}^*$); we denote by $r_c(0)$ the common input in run r. We also assume that the prover and verifier agents have initial local states $r_p(0) = s \in \{0,1\}^*$ and $r_v(0) = t \in \{0,1\}^*$, respectively, both of which contain $r_c(0)$. Additionally, we assume that nature's state at all times m includes a tuple $(\rho_p^r, \rho_v^r, \rho^r, P^*, V^*)$, where ρ_p^r and ρ_v^r are the prover's and verifier's random tapes, respectively, in run r, ρ^r is an additional tape whose role is explained in Section 3, and P^* and V^* are the protocols of the prover and verifier. An interactive protocol (P, V) generates a system. The runs of the system correspond to possible executions of (P, V). Following HMT, we denote by $P \times V$ the system consisting of all possible executions of (P, V) and by $P \times \mathcal{V}^{pp}$ the system consisting of the union of the systems $P \times V^*$ for all probabilistic, polynomial-time (PPT) protocols V^*[1]. $\mathcal{P}^{pp} \times V$ is defined analogously. More generally, we let $\mathcal{P} \times \mathcal{V}$ denote the system consisting of the union of the systems $P \times V$ for all prover protocols $P \in \mathcal{P}$ and verifier protocols $V \in \mathcal{V}$. Since we need to reason about probability, we augment a system to get a *probabilistic system*, by adding a function $\mathcal{PR}_a$ for each agent that associates with each point (r, m) a probability $\mathcal{PR}_a(r, m)$ on points for agent a, whose support is contained in $\mathcal{K}_a(r, m)$. In many cases of interest, we can think of $\mathcal{PR}_a(r, m)$ as arising from conditioning an initial probability on runs on the agent's current local state, to give a probability on points. There are subtleties to doing this though. We often do not have a probability on the set of all executions of a protocol. For example, as we observed in the case of interactive proofs, we do not want to assume a probability on the auxiliary inputs s and t or the common input x. The only source of probability is the random coin flips.

Halpern and Tuttle [1993] suggested a formalization of this intuition. Suppose that we partition the runs of $\mathcal{R}$ into cells, with a probability on each cell. For example, in the case of interactive proof systems, we could partition the runs into sets $\mathcal{R}_{s,t,x}$, according to the inputs s and t. The random coin flips of the prover and verifier protocols then give us a well-defined probability on the runs in $\mathcal{R}_{s,t}$. We can then define $\mathcal{PR}_a(r, m)$ by conditioning in the following sense: Given a set S of points, let $\mathcal{R}(S) = \{r : (r, m) \in S \text{ for some } m\}$. Let $\mathcal{R}(r)$ be the cell of the partition of $\mathcal{R}$ that includes r, and let $\Pr_{\mathcal{R}(r)}$ be the probability on the cell. If A is an arbitrary set of points, define $\mathcal{PR}_a(r, m)(A) = \Pr_{\mathcal{R}(r)}(\mathcal{R}(A \cap K_a(r, m)) \mid \mathcal{R}(\mathcal{K}_a(r, m)) \cap \mathcal{R}(r))$. (We assume for simplicity that all the relevant sets are measurable and that $\Pr_{\mathcal{R}(r)}(\mathcal{R}(\mathcal{K}_a(r, m)) \cap \mathcal{R}(r)) \neq 0$.) Note that for synchronous systems (such as those we deal with), since $\mathcal{K}_a(r, m)$ is a set of time m points, the support of $\mathcal{PR}_a(r, m)$ is a subset of time m points (i.e., $\mathcal{PR}_a(r, m)(A) = 0$ unless A includes some time m points, since otherwise $A \cap K_a(r, m) = \emptyset$). Intuitively, we associate a set of points with the set of runs going through it, and then define the probability $\mathcal{PR}_a(r, m)$, which is a's distribution on points at the point (r, m), by conditioning the probability on runs defined on r's cell on the runs going through the set $\mathcal{K}_a(r, m)$ (i.e. the runs a considers possible given his information at point (r, m)). A probabilistic system is *standard* if it is generated from probabilities on runs in this way.

In systems where the runs are generated by randomized algorithms, the cells are typically taken so as to factor out all the "nonprobabilistic" or "nondeterministic" choices. In particular, we do this for the system $P \times V$, so that we partition the runs into cells $\mathcal{R}_{s,t}$, according to the inputs s and t, as suggested above, and take the probability on the runs in the cell to be determined solely by the random inputs of the prover and verifier ρ_v and ρ_p and the random string ρ contained in nature's state. Thus, we can identify the probability on $\mathcal{R}_{s,t}$ with the uniform distribution $\Pr_{U_3}$. The probabilities on the system $\mathcal{P} \times \mathcal{V}$ are defined by the probabilities on each individual system $P \times V$ for $P \in \mathcal{P}$ and $V \in \mathcal{V}$; that is, we now partition the runs of the system into cells according to the prover and verifier protocols P, V and the inputs s and t, so there now is a separate cell for each combination of P, V, s, and t, and the probability $\Pr_{\mathcal{P} \times \mathcal{V}(r)}$ can be identified with the uniform distribution $\Pr_{U_3}$.

[1]Note that we distinguish p and v, the "prover" and the "verifier" agents respectively, from the protocols that they are running. In the system $P \times V$, the verifier is always running the same protocol V in all runs. In the system $\mathcal{P} \times \mathcal{V}^{pp}$, the verifier may be running different protocols in different runs.

2.4 Reasoning About Systems

To reason about systems, we assume that we have a collection of primitive facts such as "the value of the variable x is a prime number" (where x is the common input in the run), or "$x \in L$", where L is some set of strings. Each primitive fact φ is identified with a set $\pi(\varphi)$ of points, interpreted as the set of points at which φ holds. A point (r, m) in a system $\mathcal{R}$ satisfies φ, denoted $(\mathcal{R}, r, m) \models \varphi$, if $(r, m) \in \pi(\varphi)$. We extend this collection of primitive facts to a logical language by closing under the usual boolean connectives, the linear temporal logic operator $\Diamond$, operators $\mathtt{at\ time\ m}^*$ for each time m^*, the epistemic operators K_a, one for each agent a, and probability operators of the form for pr_a^λ each agent a and real number λ. The definitions of all these operators is standard:

- $(\mathcal{R}, r, m) \models \Diamond\varphi$ iff $(\mathcal{R}, r, m') \models \varphi$ for some $m' \geq m$.
- $(\mathcal{R}, r, m) \models K_a\varphi$ iff $(\mathcal{R}, r', m') \models \varphi$ for all $(r', m') \in \mathcal{K}_a(r, m)$. (Intuitively, agent a knows φ if φ is true at all the worlds that agent a considers possible.)
- $(\mathcal{R}, r, m) \models \mathtt{at\ time\ m}^*\ \varphi$ iff $(\mathcal{R}, r, m^*) \models \varphi$.
- $(\mathcal{R}, r, m) \models pr_a^\lambda(\varphi)$ iff $\mathcal{PR}_a(r, m)([\![\varphi]\!]) \geq \lambda$, where $[\![\varphi]\!] = \{(r', m) : (\mathcal{R}, r', m) \models \varphi\}$.

We write $\mathcal{R} \models \varphi$ if $(\mathcal{R}, r, m) \models \varphi$ for all points (r, m) in $\mathcal{R}$.

3 Characterizing Zero Knowledge Using Relation Hiding

We identify "knowing something about the initial state of the system" with "being able to generate a witness for some relation on the initial state".

For example, if the language L from which the common input x is taken is the set of all Hamiltonian graphs, then we can define a relation R such that $R(s, t, x, y)$ holds iff y is a Hamiltonian cycle in graph x. (Here the relation is independent of s and t.) Recall that a Hamiltonian cycle in a graph is a path that goes through every vertex exactly once, and starts and ends at the same vertex; a Hamiltonian graph is a graph with a Hamiltonian cycle. We can think of a Hamiltonian cycle y as a witness to a graph x being Hamiltonian. We allow the relation R to depend on s and t in addition to x because this allows us to describe the possibility of the verifier learning (via the interaction) facts about the prover's initial state (which he does not have access to). This allows us to account for provers with auxiliary information on their work tapes. For example, $R(s, t, x, y)$ could be defined to hold iff the prover has Hamiltonian path y on its work tape (in its initial state s).

We are therefore interested in relations R on $S \times T \times L \times \{0, 1\}^*$, where S is the set of prover initial states and T is the set of verifier initial states. We want a formal way to capture verifier's ability to generate such witnesses for R. We do this by using an algorithm M that takes as input the verifier's local state and the common input x, and is supposed to return a y such that $R(s, t, x, y)$ holds. The algorithm M essentially "decodes" the local state into a potential witness for R. More generally, we want to allow the decoding procedure M to depend on the protocol V^* of the verifier. We do this by using a function $\mathbf{M} : \mathcal{TM} \to \mathcal{TM}$; intuitively $\mathbf{M}(V^*)$ is the decoding procedure for the verifier protocol V^*. To reason about this in the language, we add a primitive proposition $\mathbf{M}_{v,R}$ to the language, and define $(\mathcal{R}, r, m) \models \mathbf{M}_{v,R}$ if $R(r_p(0), r_v(0), r_c(0), \mathbf{M}(V^*)(r_c(0), r_v(m))(\rho^r))$ holds, where V^* is the verifier protocol in run r and ρ^r is the extra random tape that is part of nature's local state in run r; this makes the output of $\mathbf{M}(V^*)$ in run r deterministic (although M is a probabilistic TM). For any constant λ, let $G_v^{\mathbf{M},m^*,\lambda}R$, read "the verifier can generate a y satisfying R using $\mathbf{M}$ with probability λ at time m^*" be an abbreviation of $pr_v^\lambda(\mathtt{at\ time\ m}^*\ \mathbf{M}_{v,R})$. We can generalize this to a formula $G_v^{\mathbf{M},m^*,\lambda}R$ which considers functions λ whose meaning may depend on components of the state, such as the verifier's protocol and the length of the common input; we leave the straightforward semantic details to the reader. $G_p^{\mathbf{M},m^*,\lambda}R$, read "the prover can generate a y satisfying R using $\mathbf{M}$ with probability λ at time m^*", is defined analogously. Finally, we add the primitive proposition $\mathtt{s} \in \mathtt{R_L}(\mathtt{x})$ to the language, and define $(\mathcal{R}, r, m) \models \mathtt{s} \in \mathtt{R_L}(\mathtt{x})$ if $r_c(0) \in L$ and $r_p(0) \in R_L(r_c(0))$.

We now show how to use the formula $G_v^{\mathbf{M},m^*,\lambda}R$ to capture the intuitions underlying zero-knowledge proofs. Intuitively, we want to say that if the verifier can generate a y satisfying a relation R after the interaction, he could also do so before the interaction (i.e., without interacting with the prover at all). However, this is not quite true; a verifier can learn a y satisfying R during the course of an interaction, but only in a negligibly small fraction of the possible conversations. We want to capture the fact that the probability of the verifier being able to generate the witness correctly at a final point in the system is only negligibly different from the probability he can do so at the corresponding initial point (in a perfect zero knowledge system, the probabilities are exactly the same). Note that when the Turing machine M used by the verifier in a particular run r generates a y, the verifier may not know whether y in fact is a witness; that is, the verifier may not know whether $R(s, t, x, y)$ in fact holds. Nevertheless, we want it to be the case that if the verifier can use some algorithm M that generates a witness y with a certain probability after interacting with the prover, then the verifier can generate a witness y with the same probability without the interaction. This lets us account for leaks in knowledge from the interaction that the verifier may not be aware of. For example, a computationally bounded verifier may have a Hamiltonian cycle y in graph x as part of his local state, but no way of knowing that y is in fact a Hamiltonian cycle. We want to say that the verifier knows how to generate a Hamiltonian cycle if this is the case (even if he does not know that he can do so), since there is a way for the verifier to extract a Hamiltonian cycle from his local state.

We now define relation hiding, which says that if the verifier initially knows that he can, at some future time during the interaction with the prover, generate a witness for some relation R on the initial state with some probability, then he knows that he can generate a witness for R at time 0, that is, before the interaction, with almost the same probability. We allow the generating machines used by the verifier (both after and before the interaction) to run in *expected* polynomial time in the common input and verifier view. Allowing them to only run in (strict) polynomial time, would certainly also be a reasonable choice, but this would result in a notion that is stronger than the traditional notion of zero-knowledge.[2] Let $\mathcal{EPPT}$ be the set of all expected probabilistic polynomial time algorithms (i.e., algorithms for which there exists a polynomial p such that the expected running time on input x is at most $p(|x|)$).

Definition 3 *The system $\mathcal{R}$ is* relation hiding for L *if, for every polynomial-time relation R on $S \times T \times L \times \{0,1\}^*$ and function $\mathbf{M} : \mathcal{TM} \to \mathcal{EPPT}$, there exist functions $\mathbf{M}' : \mathcal{TM} \to \mathcal{EPPT}$, $\epsilon : \mathcal{TM} \times \mathbb{N} \to [0,1]$ such that for every Turing machine V^*, $\epsilon(V^*, \cdot)$ is a negligible function, and for every $0 \le \lambda \le 1$ and time m^*,*

$$\mathcal{R} \models \mathtt{at\ time\ 0}\ (\mathtt{s} \in \mathtt{R_L}(\mathtt{x}) \land G_v^{\mathbf{M},m^*,\lambda}R \Rightarrow G_v^{\mathbf{M}',0,\lambda-\epsilon}R).$$

[2] In fact, it would result in a notion called *strict polynomial-time zero knowledge* [Goldreich 2001].

In Definition 3, we allow the meaning of ϵ to depend on the verifier's protocol V^* since, intuitively, different verifier protocols may result in different amounts of knowledge being leaked. If we had not allowed ϵ to depend on the verifier protocol V^*, we would need a single negligible function that bounded the "leakage" of information for all verifiers in $\mathcal{V}^{pp}$. We cannot prove that such a function exists with the traditional definition of zero knowledge. Similarly, we must allow $\mathbf{M}'$ to depend on the verifier's protocol, even if $\mathbf{M}$ does not. Intuitively, $\mathbf{M}'$ must be able to do at time 0 what $\mathbf{M}$ can do at time m^*, so it must know something about what happened between times 0 and m^*. The verifier's protocol serves to provide this information, since for each verifier protocol V^*, the definition of zero knowledge ensures the existence of a simulator S_{V^*} that can be used to mimic the interaction before time m^*. The relation-hiding property captures the requirement that if the verifier can eventually generate an arbitrary R, he can do so almost as well (i.e. with negligibly lower probability of correctness) initially. We now use this property to characterize zero knowledge.

Theorem 1 *The interactive proof system (P, V) for L is computational zero knowledge iff the system $P \times \mathcal{V}^{pp}$ is relation hiding for L.*

Theorem 1 says that if (P, V) is a computational zero-knowledge proof system, then for any PPT verifier and relation R, if the verifier can eventually generate a witness for R, he can do so almost as well initially. Note that in this characterization of zero knowledge, the prover does not need to know the verifier's protocol to know that the statement holds. An intuition for the proof of Theorem 1 follows: the details (as well as all other proofs) can be found at www.cs.cornell.edu/home/halpern/papers/tark09a.pdf.

For the "if" direction, suppose that (P, V) is a computational zero knowledge system. If V^* is the verifier protocol in run $r \in P \times \mathcal{V}^{pp}$, then there is a simulator machine S_{V^*} that produces verifier views that no distinguisher D can distinguish from views during possible interactions with the prover, no matter what auxiliary input D has. We show that if the verifier has an algorithm $\mathbf{M}(V^*)$ that takes as input his view at a final point of the interaction and generates a y satisfying the relation R, then he can generate such a y before the interaction by running the simulating machine S_{V^*} at the initial point to get a final view, and then running $\mathbf{M}(V^*)$ on this view to generate y. We can therefore construct the function $\mathbf{M}'$ using $\mathbf{M}$ and S_{V^*}.

For the "only if" direction, given an arbitrary protocol V^*, we construct a relation R such that the verifier has an algorithm for generating witnesses for R after the interaction. Since $P \times \mathcal{V}^{pp}$ is relation hiding for L, the verifier has an algorithm for generating witnesses for R at initial points of the interaction. We then use this generating machine to implement a simulator S_{V^*} that fools any distinguisher.

4 Characterizing Variants of Zero Knowledge

We can use the ideas of relation hiding to characterize variants of zero knowledge. In this section, we show how to characterize two well-known variants: concurrent common knowledge and proofs of knowledge.

4.1 Concurrent Zero Knowledge

So far, we have considered only single executions of an interactive proof system. However, zero-knowledge proofs are often used in the midst of other protocols. Moreover, when this is done, several zero-knowledge proofs may be going on concurrently. An adversary may be able to pass messages between various invocations of zero-knowledge proofs to gain information. Dwork, Naor, and Sahai [1998] presented a definition of *concurrent* zero knowledge that tries to capture the intuition that no information is leaked even in the presence of several concurrent invocations of a zero-knowledge protocol. They consider a probabilistic polynomial-time verifier that can talk to many independent provers (all using the same protocol) concurrently. The verifier can interleave messages to and from different provers as desired. We say that an *extended verifier protocol* is a protocol for the verifier where the verifier can interact with arbitrarily many provers concurrently, rather than just one prover. (Since we are interested in verifiers that run in polynomial time, for each extended verifier protocol V there is a polynomial q_V such that the verifier can interact with only $q_V(|x|)$ provers on input x. This means that the verifier's view also contains messages to and from at most $q_V(|x|)$ provers.) Denote by $(\tilde{P}(s) \leftrightarrow V(t))(x)$ the random variable that takes an infinite tuple of infinite random strings $((\rho_{p_i})_{i\in\mathbb{N}}, \rho_v)$ as input and outputs an execution where all the provers are running protocol P with auxiliary input s on common input x and the verifier is running the extended verifier protocol V with auxiliary input t and common input x, prover i has the infinite string ρ_i on its random tape, and the verifier has ρ_v on its random tape.

With this background, we can define a concurrent definition of zero knowledge in exactly the same way as zero knowledge (Definition 2), except that we now consider extended verifier protocols; we omit the details here.

We can model a concurrent zero-knowledge system in the runs and systems framework as follows. We now consider systems with an infinite number of agents: a verifier v and an infinite number of provers $p_1, p_2, \ldots$. All agents have common input $r_c(0)$ in run r. As before, the provers and the verifier have initial local states. We will be interested in systems where all the provers have the same initial state and use the same protocol. Moreover, this will be a protocol where a prover talks only to the verifier, so the provers do not talk to each other. This captures the fact that the verifier can now talk to multiple provers running the same protocol, but the provers themselves cannot interact with each other (they are independent). Again, the initial local states of the provers and the verifier all contain $r_c(0)$. Additionally, we assume that nature's state at all times m includes a tuple $(\rho^r_{p_1}, \ldots, \rho^r_v, \rho^r, P^*, V^*)$, where $\rho^r_{p_i}$ is prover p_i's random tape and ρ^r_v is the verifier's random tape in run r, ρ^r is an additional tape as before, P^* is the protocol used by all the provers, and V^* is the verifier's protocol. Note that the provers' random tapes are all independent to ensure that their actions are not correlated. Given a set $\mathcal{P}$ of prover protocols and $\mathcal{V}$ of verifier protocols, let $\tilde{\mathcal{P}} \times \mathcal{V}$ denote the system with runs of this form, where the provers' protocol is in $\mathcal{P}$ and the verifier's protocol in $\mathcal{V}$. If $\mathcal{P} = \{P\}$, we write $\tilde{P} \times \mathcal{V}$. We define the probability on $\tilde{\mathcal{P}} \times \mathcal{V}$ as before, partitioning the runs into cells according to the protocol used and the inputs. Thus, we can identify the probability on $\mathcal{R}_{s,t,P,V}$ with the uniform distribution $\Pr_{U_\infty}$.

Theorem 2 *The interactive proof system (P, V) for L is computational concurrent zero knowledge iff the system $\tilde{P} \times \mathcal{V}^{pp}$ is relation hiding for L.*

The proof is almost identical to that of Theorem 1.

4.2 Proofs of Knowledge

Loosely speaking, an interactive proof is a proof of knowledge if the prover not only convinces the verifier of the validity of some statement, but also that it possesses, or can "feasibly compute", a witness for the statement proved (intuitively, using the secret information in its initial state). For instance, rather than merely convincing the verifier that a graph is Hamiltonian, the prover convinces the verifier that he knows a Hamiltonian cycle in the graph. We show how this notion can be formalized using our logic. There are a number of ways of formalizing proofs of knowledge; see, for example, [Bellare and Goldreich 1992; Feige, Fiat, and Shamir 1987; Feige and Shamir 1990; Tompa and Woll 1987]. We give here a definition that is essentially that of Feige and Shamir [1990]. In the full paper, we discuss modifications that give the other variants, and how to modify our characterization to handle them.

Definition 4 *An interactive proof system (P, V) for a language L with witness relation R_L is a* proof of knowledge *if, for every PPT prover protocol P^*, there exists a negligible function ϵ and a probabilistic Turing machine E_{P^*} that takes as input the common input x and prover's auxiliary information s, runs in expected time polynomial in $|x|$, and outputs a candidate witness for x, such that for all $x, s, t \in \{0,1\}^*$,*

$$\Pr_U[\{Accepts_v[(P(s) \leftrightarrow V(t))(x)]\}] - \Pr_U[\{E_{P^*}(x, s)) \in R_L(x)\}] \leq \epsilon(|x|).$$

Intuitively, this says that for every prover P^*, if P^* succeeds in convincing the verifier V that x is in L, then there is a "knowledge extractor" machine E_{P^*} that can extract a witness for x from the prover's auxiliary information. We can think of the extractor as demonstrating that the prover really does know a witness to show that $x \in L$, given its auxiliary information s. We now formalize this definition of proofs of knowledge using our logic. Let `accepts` denote the primitive proposition that holds iff the verifier enters an accept state at the end of the interaction.

Definition 5 *The system $\mathcal{R}$ is* witness convincing for the language L with witness relation R_L *if there exist functions $\mathbf{M} : \mathcal{TM} \to \mathcal{EPPT}$, $\epsilon : \mathcal{TM} \times \mathbb{N} \to [0,1]$ such that, for every Turing machine P^*, $\epsilon(P^*, \cdot)$ is a negligible function, and, for all $0 \leq \lambda \leq 1$,*

$$\mathcal{R} \models \texttt{at time 0}\ pr_p^{\lambda}(\Diamond \texttt{accepts}) \Rightarrow G_p^{\mathbf{M},0,\lambda-\epsilon} R_L^+,$$

where $(s, t, x, y) \in R_L^+$ iff $y \in R_L(x)$.

This definition says that there exists a function $\mathbf{M}$ such that $\mathbf{M}(P^*)$ can generate a y such that $(s, t, x, y) \in R_L^+$ whenever P^* makes the verifier accept in the system $\mathcal{R}$. This machine can be viewed as a knowledge extractor for P^*, motivating the following theorem.

Theorem 3 *The interactive proof system (P, V) for L is a proof of knowledge iff the system $\mathcal{P}^{pp} \times V$ is witness convincing for L.*

To see why this should be true, note that if (P, V) is a proof of knowledge and if the verifier accepts on input x when interacting with P^*, then there exists a knowledge extractor machine E_{P^*} that can generate a witness $y \in R_L(x)$, and can therefore generate a y such that $(s, t, x, y) \in R_L^+$. For the converse, as we said above, the machine $\mathbf{M}(P^*)$ that exists by the definition of witness convincing can be viewed as a knowledge extractor for P^*. Again, the details can be found at www.cs.cornell.edu/home/halpern/papers/tark09a.pdf.

The major difference between the FFS and TW difference is that, in the FFS definition, rather than allowing a different machine E_{P^*} for every prover protocol P^*, FFS require that there be a single knowledge extractor machine E that has oracle access to the prover's protocol and a fixed bivariate polynomial p such that the running time of E given a prover P^* with runtime bounded by a polynomial q and input x is $p(q(|x|, |x|)$. To capture this difference. we vary the definition of witness convincing to require that $\mathbf{M}(\mathbf{P}^*)$ for any P^* return the same machine M that takes (a description of) P^* as an input and has expected runtime polynomial in the runtime of P^* and $|x|$.

5 Conclusions and Future Work

HMT formalized the notion of *knowledge security* and showed that a zero-knowledge proof system for $x \in L$ satisfies it: the prover is guaranteed that, with high probability, if the verifier will practically know (as defined in [Moses 1988]) a fact φ at the end of the proof, he practically knows $x \in L \Rightarrow \varphi$ at the start. They also formalized the notion of knowing how to generate a y satisfying any relation $R(x, y)$ that is BPP-testable by the verifier, and showed that zero-knowledge proofs also satisfy the analogous property of generation security (with respect to these relations). Their work left open the question of whether either of these notions of security characterizes zero knowledge.

We have provided a different definition of what it means for a polynomial-time agent to know how to generate a string y satisfying a relation R. Using this definition we provide a logical statement—called relation hiding—that fully characterizes when an interaction is zero knowledge. We additionally show that variants of this statement (using the same notion of knowing how to generate) characterize variants of zero knowledge, including concurrent zero knowledge and proofs of knowledge.

Our notion of relation hiding considers the verifier's knowledge at the beginning of a run (i.e. at time 0); it says that, at time 0, the verifier cannot know that he will be able to generate a witness for a relation with higher probability in the future than he currently can. We would like to make the stronger claim that the verifier will *never* know that he can generate a witness satifying the relation better than he knows he can at the beginning (or, more accurately, will almost certainly never know this, since there is always a negligible probability that he will learn something). To do this, we need to talk about the verifier's knowledge and belief at all points in the system. Consider, for example, a verifier trying to factor a large number. We would like to allow for the fact that the verifier will, with some small probability, get the correct answer just by guessing. However, we want to be able to say that if, after interacting with the prover, the verifier believes that he can guess the factors with non-negligible probability then, except with very small probability, he already believed that he could guess the factors with almost the same probability before the interaction. Making this precise seems to require some axioms about how a computationally-bounded verifier's beliefs evolve. We are currently working on this, using Rantala's "impossible possible-worlds approach" [Rantala 1982] to capture the verifiers computational bounds. For example, if the verifier cannot compute whether a number n is prime, he will consider possible a world where n is prime and one where it is not (although one of these worlds is logically impossible). Proving analogues of our theorem in this setting seems like an interesting challenge, which will lead to a deeper understanding of variants of zero knowledge.

A Proofs

A.1 Computational Zero Knowledge

Theorem 1: *The interactive proof system (P, V) for L is computational zero knowledge iff the system $P \times \mathcal{V}^{pp}$ is relation hiding for L.*

Proof. For the "if" direction, suppose that (P, V) is a computational zero knowledge system and that $(P \times \mathcal{V}^{pp}, r, 0) \models G_v^{\mathbf{M},\mathbf{m}^*,\lambda} R$ for a polynomial-time relation R and functions $\mathbf{M} : \mathcal{TM} \to \mathcal{EPPT}$ and $\lambda : \mathcal{TM} \times \mathbb{N} \to [0, 1]$. We want to show that there exist functions $\mathbf{M}' : \mathcal{TM} \to \mathcal{EPPT}$ and $\epsilon : \mathcal{TM} \times \mathbb{N} \to [0, 1]$ such that $(P \times \mathcal{V}^{pp}, r, 0) \models G_v^{\mathbf{M}',0,\lambda-\epsilon} R$. The intuition behind the proof is as follows. If (P, V) is zero knowledge, and V^* is the verifier protocol in run r, then there is a simulator machine S_{V^*} that produces verifier views that no distinguisher D can distinguish from views during possible interactions with the prover, no matter what auxiliary input D has. We show that if the verifier has an algorithm $\mathbf{M}(V^*)$ that takes as input his view at a final point of the interaction and generates a y satisfying the relation R, then he can generate such a y before the interaction by running the simulating machine S_{V^*} at the initial point to get a final view, and then running $\mathbf{M}(V^*)$ on this view to generate y. We can therefore construct the function $\mathbf{M}'$ using $\mathbf{M}$ and S_{V^*}.

In more detail, we want to show $(P \times \mathcal{V}^{pp}) \models \mathtt{at\ time\ 0}\ (\mathtt{s} \in \mathtt{R_L(x)} \wedge G_v^{\mathbf{M},m^*,\lambda} R \Rightarrow G_v^{\mathbf{M}',0,\lambda-\epsilon} R)$. Thus, we must show that for all runs r, we have $(P \times \mathcal{V}^{pp}, r, 0) \models (\mathtt{s} \in \mathtt{R_L(x)} \wedge G_v^{\mathbf{M},m^*,\lambda} R \Rightarrow G_v^{\mathbf{M}',0,\lambda-\epsilon} R)$. So suppose that $r_c(0) \in L$, $r_p(0) \in R_L(r_c(0))$, and $(\{P\} \times \mathcal{V}^{pp}, r, 0) \models G_v^{\mathbf{M},m^*,\lambda} R$. By definition, this means that $(P \times \mathcal{V}^{pp}, r, 0) \models pr_v^{\lambda}(\mathtt{at\ time\ m^*}\ \mathbf{M}_{v,R})$. Assume without loss of generality that m^* is greater than the final time of the interaction in all runs with input x. (There is such an m^*, since V^* runs in time polynomial in $|x|$. This assumption is made without loss of generality since we are assuming perfect recall, so anything the verifier can do in the middle of the interaction, he can do at the end). Construct a PPT distinguisher D as follows. D takes as input a verifier view $view_v$ and extracts the verifier's initial state t from it (since by perfect recall, the initial verifier state is contained in any subsequent view). Recall that distinguishers can take auxiliary inputs as well as a view. In this case, in run r we choose to give D as auxiliary input the common input x and the prover's state s in r. Given x and s and a random string ρ, D runs $\mathbf{M}(V^*)$ on x, $view_v$, and ρ, where V^* is the verifier's protocol in run r, to get y, and outputs $R(s, t, x, y)$. So D with inputs x, s, and ρ accepts the verifier's view $view_v$ iff $R(s, t, x, \mathbf{M}(V^*)(view_v)(\rho)) = 1$ for the t contained in the verifier's view.

Suppose that the verifier runs V^* in r. Since $(P \times \mathcal{V}^{pp}, r, 0) \models pr_v^{\lambda}(\mathtt{at\ time\ m^*}\ \mathbf{M}_{v,R})$, we have

$$\mathcal{PR}_v(r,0)(\{(r',0) : (P\times V^*, r', 0) \models \mathtt{at\ time\ m^*}\ \mathbf{M}_{v,R}\}, r'_v(0) = r_v(0), r'_p(0) = r_p(0)\}) \geq \lambda(V^*, |r_c(0)|).$$

Recall that we can identify $\mathcal{PR}_v(r, 0)$ with $\Pr_{U_3}$, so

$$(\Pr_U(\{r' \in (\mathcal{P}\times\mathcal{V})(r) : (P\times V^*, r', 0) \models \mathtt{at\ time\ m^*}\ \mathbf{M}_{v,R}, r'_v(0) = r_v(0), r'_p(0) = r_p(0)\}) \geq \lambda(V^*, |r_c(0)|).$$

By definition of $\mathbf{M}_{v,R}$,

$$\Pr_U[\{r' \in (\mathcal{P} \times \mathcal{V})(r) : R(r'_p(0), r'_v(0), r'_c(0), \mathbf{M}(V^*)(r'_c(0), r'_v(m^*))) = 1\}] \geq \lambda(V^*, |r_c(0)|).$$

By definition of $(\mathcal{P} \times \mathcal{V})(r)$,

$$\Pr_U[R(r_p(0), r_v(0), r_c(0), \mathbf{M}(V^*)(r_c(0), r_v(m^*))) = 1] \geq \lambda(V^*, |r_c(0)|).$$

Thus, D with auxiliary inputs $r_c(0)$ and $r_p(0)$ accepts the verifier's view $r(m^*)$ with probability at least $\lambda(V^*, |r_c(0)|)$ (where the probability is taken over the random choices of D).

Because (P, V) is a computational zero-knowledge proof system for L, if $r_c(0) \in L$ and $r_p(0) \in R_L(r_c(0))$, then there is an expected PPT Turing machine S_{V^*} and a negligible function $\epsilon(V^*)$ such that S_{V^*} on input $r_c(0), r_v(0)$ outputs a verifier view such that every distinguisher, and in particular the distinguisher D constructed above, accepts this view with probability at least $(\lambda - \epsilon)(V^*, |r_c(0)|)$ (taken over the random choices of D and S_{V^*}), no matter what auxiliary information we give it, and in particular given auxiliary inputs $r_c(0)$ and $r_p(0)$. Thus, by the definition of D, we must have

$$\Pr_U[R(r_p(0), r_v(0), r_c(0), \mathbf{M}(V^*)(r_c(0), r_v(m^*))) = 1] \geq \lambda(V^*, |r_c(0)|).$$

Define $\mathbf{M}' : \mathcal{TM} \to \mathcal{EPPT}$ by taking $\mathbf{M}'(V^*)(x,t) = \mathbf{M}(V^*)(x, S_{V^*}(x,t))$. Note that this definition suppresses the random choices of $\mathbf{M}'(V^*)$, S_{V^*} and $\mathbf{M}(V^*)$—we assume that each of these machines is given a random tape, and that the random tapes of S_{V^*} and $\mathbf{M}(V^*)$ are independent, so that their outputs are not correlated. Since S_{V^*} and $\mathbf{M}(V^*)$ are both expected polynomial-time in $|x|$ and $|t|$, so is $\mathbf{M}'(V^*)$. Note also that

$$R(r_p(0), r_v(0), r_c(0), \mathbf{M}'(V^*)(r_c(0), r_v(0))(\rho^r)) = 1$$

iff

$$R(r_p(0), r_v(0), r_c(0), \mathbf{M}(V^*)(r_c(0), S_{V^*}(r_c(0), r_v(0))(\rho_v^r))(\rho^r)) = 1;$$

thus,

$$\Pr_U[R(r_p(0), r_v(0), r_c(0), \mathbf{M}'(V^*)(r_c(0), r_v(0))) = 1] \geq (\lambda - \epsilon)(V^*, |r_c(0)|).$$

So $\mathbf{M}'(V^*)$ runs in expected polynomial time and outputs a value such that

$$(P \times \mathcal{V}^{pp}, r, 0) \models pr_v^{\lambda - \epsilon}(\texttt{at time 0 } \mathbf{M}'_{v,R}).$$

This completes the proof of the "if" direction.

For the "only if" direction, we want to show that for every verifier protocol V^*, there is an EPPT algorithm S_{V^*} such that for any PPT distinguisher D with any auxiliary input, there exists a negligible function ϵ such that $\forall x \in L, \forall s \in R_L(x), t, z \in \{0,1\}^*$,

$$|\Pr_U[\{D(S_{V^*}(x,t), z) = 1\}] - \Pr_U[\{D(View_v[(P(s) \leftrightarrow V^*(t))(x)], z) = 1\}]| \leq \epsilon(|x|).$$

The idea behind the proof is as follows. Given an arbitrary protocol V^*, we construct a relation R such that the verifier has an algorithm for generating witnesses for R after the interaction. Since $P \times \mathcal{V}^{pp}$ is relation hiding for L, the verifier has an algorithm for generating witnesses for R at initial points of the interaction. We then use this generating machine to implement a simulator S_{V^*} that fools any distinguisher.

Recall that the set of possible verifier initial states is the set of all bitstrings $\{0,1\}^*$. For an arbitrary PPT distinguisher D, define the set $T_D \subseteq \{0,1\}^*$ of verifier states of the form $t = t'; 1^{poly(|x|)}; D; z; \rho_D$, where $t' \in \{0,1\}^*$, $x \in L$, D is the description of the PPT distinguisher, $z \in \{0,1\}^*$, $\rho_D \in \{0,1\}^*$, and $poly(\cdot)$ is some polynomial. Define a relation R by taking $R(s,t,x,y) = 1$ iff t is of the form $t = t'; 1^{poly(|x|)}; D; z; \rho_D$ and the distinguisher D contained in t accepts on input y when given the random string ρ_D and auxiliary input z contained in t (otherwise $R(s,t,x,y) = 0$).

Define $\mathbf{M}(V^*)$ to be the trivial machine that on input $(r_c(0), r_v(m))$ outputs the verifier's view $r_v(m)$. Suppose that there exists a function λ and a set A of runs such that for all $r \in A$, if m_r^* is the final round in run r, then $(P \times \mathcal{V}^{pp}, r, 0) \models G_v^{\mathbf{M}, m_r^*, \lambda} R$. Since $P \times \mathcal{V}^{pp}$ is relation hiding for

L, there exists a function $\mathbf{M}' : \mathcal{TM} \to \mathcal{EPPT}$ and a function $\epsilon : \mathcal{TM} \times \mathbb{N}$ such that $\epsilon(V^*)$ is negligible and for all $r \in A$, $(P \times \mathcal{V}^{pp}, r, 0) \models \mathtt{s} \in \mathtt{R_L(x)} \Rightarrow G_v^{\mathbf{M}',0,\lambda-\epsilon} R$. Define S_{V^*} by taking $S_{V^*}(x,t)(\rho) = \mathbf{M}'(V^*)(x,t)(\rho)$.

Suppose, by way of contradiction, that for some distinguisher D there is a polynomial p such that for infinitely many

$x \in L$, $s \in R_L(x)$, and $t \in \{0,1\}^*$, there is a $z \in \{0,1\}^*$ such that there exist functions λ_1^z and λ_2^z (that may depend on z) such that $\Pr_U[\{D(S_{V^*}(x,t), z) = 1\}] = \lambda_1^z(x,s,t)$, $\Pr_U[\{D(View_v[(P(s) \leftrightarrow V^*(t))(x)], z) = 1\}] = \lambda_2^z(x,s,t)$, and $|\lambda_1^z(x,s,t) - \lambda_2^z(x,s,t)| > \frac{1}{p(|x|)}$. Consider the set $T_{D,S_{V^*}} \subseteq T_D$ of verifier states of the form $t = t'; 1^{q(|x|)}; D; z; \rho_D$, where q is a polynomial such that $q(|x|)$ is an upper bound on the running time of the verifier protocol V^* and the simulator S_{V^*} on input x. The effect of this choice is that, given the verifier's state, the verifier and simulator cannot access the distinguisher description. Therefore, $View_v[(P(s) \leftrightarrow V^*(t))(x)(\rho_p, \rho_v)] = View_v[(P(s) \leftrightarrow V^*(t'))(x)(\rho_p, \rho_v)]$ and $S_{V^*}(x,t)(\rho_v) = S_{V^*}(x,t')(\rho_v)$. If there exists $z \in \{0,1\}^*$ such that the distinguisher D, given auxiliary input z, succeeds in distinguishing the distributions $\{View_v[(P(s) \leftrightarrow V^*(t'))(x)\}$ and $\{S_{V^*}(x,t')\}$, then D can distinguish $\{View_v[(P(s) \leftrightarrow V^*(t))(x)\}$ and $\{S_{V^*}(x,t)\}$ given z. By construction of $T_{D,S_{V^*}}$, for infinitely many $x \in L, s \in R_L(x)$ and $t \in T_{D,S_{V^*}}$, there exist functions λ_1^z and λ_2^z such that $\Pr_U[\{D(S_{V^*}(x,t), z) = 1\}] = \lambda_1^z(x,s,t)$, $\Pr_U[\{D(View_v[(P(s) \leftrightarrow V^*(t))(x)], z) = 1\}] = \lambda_2^z(x,s,t)$, and $|\lambda_1^z(x,s,t) - \lambda_2^z(x,s,t)| > \frac{1}{p(|x|)}$ (the z referenced here is contained in t). Without loss of generality, we can assume that $\lambda_2^z(x,s,t) - \lambda_1^z(x,s,t) > \frac{1}{p(|x|)}$ for infinitely many of the relevant x's, s's, and t's. To see that this is without loss of generality, note that if $\lambda_2^z(x,s,t) - \lambda_1^z(x,s,t) > \frac{1}{p(|x|)}$ for only a finite number of x's, s's, and t's, then it must be the case that $\lambda_1^z(x,s,t) - \lambda_2^z(x,s,t) > \frac{1}{p(|x|)}$ for infinitely many x's, s's, and t's. In this case, we can define a distinguisher D' that outputs the opposite of what D outputs when given the same view and auxiliary input. Then for infinitely many $x \in L$, s, and t, there exists $z \in \{0,1\}^*$ such that $\Pr_U[\{D'(S_{V^*}(x,t), z) = 1\}] = \lambda_1'^z(x,s,t)$, $\Pr_U[\{D'(View_v[(P(s) \leftrightarrow V^*(t))(x)], z) = 1\}] = \lambda_2'^z(x,s,t)$, and $\lambda_2'^z(x,s,t) - \lambda_1'^z(x,s,t) > \frac{1}{p(|x|)}$ where $\lambda_1'^z = 1 - \lambda_1^z$ and $\lambda_2'^z = 1 - \lambda_2^z$. We can then proceed with the rest of the proof using the distinguisher D' instead of D.

Let A denote the set of tuples (x,s,t) (with $x \in L, s \in R_L(x), t \in T_{D,S_{V^*}}$) for which there exists $z \in \{0,1\}^*$ and functions λ_1^z, λ_2^z such that

$\Pr_U[\{D(S_{V^*}(x,t), z) = 1\}] = \lambda_1^z(x,s,t)$, $\Pr_U[\{D(View_v[(P(s) \leftrightarrow V^*(t))(x)], z) = 1\}] = \lambda_2^z(x,s,t)$, and $\lambda_2^z(x,s,t) - \lambda_1^z(x,s,t) > \frac{1}{p(|x|)}$. In the system $P \times \mathcal{V}^{pp}$, let $A' = \{r \in P \times V^* : (r_c(0), r_p(0), r_v(0)) \in A\}$. So for all $r \in A'$, $\Pr_U[\{D(S_{V^*}(r_c(0), r_v(0)), z) = 1\}] = \lambda_1^z(r_c(0), r_p(0), r_v(0))$ (where z is contained in $r_p(0)$), $\Pr_U[\{D(View_v[(P(r_p(0)) \leftrightarrow V^*(r_v(0)))(r_c(0))], z) = 1\}] = \lambda_2^z(r_c(0), r_p(0), r_v(0))$, and $\lambda_2^z(r_c(0), r_p(0), r_v(0)) - \lambda_1^z(r_c(0), r_p(0), r_v(0)) > \frac{1}{p(|r_c(0)|)}$.

So for all $r \in A'$, if m_r^* is the final round in run r, then by definition of R, $\mathbf{M}$, and $\mathbf{M}'$,

$$\Pr_U[R(r_p(0), r_v(0), r_c(0), \mathbf{M}(V^*)(r_c(0), r_v(m_r^*)))] = \lambda_2^z(r_c(0), r_p(0), r_v(0)),$$

$$\Pr_U[R(r_p(0), r_v(0), r_c(0), \mathbf{M}'(V^*)(r_c(0), r_v(0)))] = \lambda_1^z(r_c(0), r_p(0), r_v(0)),$$

and

$$\lambda_2^z(r_c(0), r_p(0), r_v(0)) - \lambda_1^z(r_c(0), r_p(0), r_v(0)) > \frac{1}{p(|r_c(0)|)}.$$

So for any negligible function $\epsilon(V^*)$,

$$\Pr_U[R(r_p(0), r_v(0), r_c(0), \mathbf{M}'(V^*)(r_c(0), r_v(0)))] < \lambda_2^z(r_c(0), r_p(0), r_v(0)) - \epsilon(V^*)(|x|)$$

for all but finitely many x. By definition of pr_a^λ, for all $r \in A'$,

$$(P \times \mathcal{V}^{pp}, r, 0) \models pr_v^{\lambda_2^z}(\texttt{at time m}_\texttt{r}^*\,)\mathbf{M}_{v,R}$$

and

$$(P \times \mathcal{V}^{pp}, r, 0) \not\models pr_v^{\lambda_\mathbf{2}^\mathbf{z}-\epsilon}(\texttt{at time 0}\,)\mathbf{M}'_{v,R}$$

for any negligible function $\epsilon(V^*)$. Also, by definition of A', $(P \times \mathcal{V}^{pp}, r, 0) \models \texttt{s} \in \texttt{R}_\texttt{L}(\texttt{x})$, so

$$(P \times \mathcal{V}^{pp}, r, 0) \models \texttt{s} \in \texttt{R}_\texttt{L}(\texttt{x}) \wedge G_v^{\mathbf{M}, m_r^*, \lambda_\mathbf{2}^\mathbf{z}} R$$

and

$$(P \times \mathcal{V}^{pp}, r, 0) \not\models G_v^{\mathbf{M}', 0, \lambda_\mathbf{2}^\mathbf{z}-\epsilon} R$$

for any function $\epsilon : \mathcal{TM} \times \mathbb{N} \to [0, 1]$ such that $\epsilon(V^*)$ is negligible. This gives us a contradiction. ∎

A.2 Concurrent Zero Knowledge

Theorem 2: *The interactive proof system (P, V) for L is computational concurrent zero knowledge iff the system $\tilde{P} \times \mathcal{V}^{pp}$ is relation hiding for L.*

Proof. For the "if" direction, suppose that (P, V) is a computational concurrent zero knowledge system and that $(\tilde{P} \times \mathcal{V}^{pp}, r, 0) \models G_v^{\mathbf{M}, \mathbf{m}^*, \lambda} R$ for some arbitrary polynomial-time relation R and some functions $\mathbf{M} : \mathcal{TM} \to \mathcal{EPPT}$, $\lambda : \mathcal{TM} \times \mathbb{N} \to [0, 1]$. We want to show that there exist functions $\mathbf{M}' : \mathcal{TM} \to \mathcal{EPPT}$, $\epsilon : \mathcal{TM} \times \mathbb{N} \to [0, 1]$ such that $(\tilde{P} \times \mathcal{V}^{pp}, r, 0) \models G_v^{\mathbf{M}', 0, \lambda - \epsilon} R$.

Let V^* be the extended verifier protocol in run $r \in \tilde{P} \times \mathcal{V}^{pp}$, let x be the common input, and let $q_{V^*}(|x|)$ be an upper bound on the runtime of V^* on common input x. Recall that the verifier's local state at time $m \geq 0$ consists of the time on the global clock, his initial information $r_v(0)$ (which contains the common input $r_c(0)$), the history of messages he has received from other agents and read, and the history of coin flips he has used. Since the verifier's running time is bounded by $q_{V^*}(|r_c(0)|)$, no matter how many messages he receives in round m, he can read at most $q_{V^*}(|r_c(0)|)$ of them. So his local state (and his view) at any time $m \geq 0$ can contain at most $q_{V^*}(|r_c(0)|)$ messages, coming from at most $q_{V^*}(|r_c(0)|)$ provers (indexed without loss of generality by $1, 2, \ldots, q_{V^*}(|r_c(0)|)$). So for every run r in $\tilde{P} \times \mathcal{V}^{pp}$, the verifier interacts with a subset of $p_1, p_2, \ldots, p_{q_{V^*}(|r_c(0)|)}$. By the definition of concurrent zero knowledge, there is a simulator machine S_{V^*} such that S_{V^*} produces verifier views that are indistinguishable by any distinguisher (with any auxiliary input) from views during possible interactions of V^* with up to $q_{V^*}(|x|)$ instances of P on common input $|x|$. The proof that $\tilde{P} \times \mathcal{V}^{pp}$ is relation hiding now proceeds exactly as in Theorem 1.

For the "only if" direction, we want to show that for every verifier protocol V^*, there is an EPPT algorithm S_{V^*} such that for any PPT distinguisher D with any auxiliary input, there exists a negligible function ϵ such that $\forall x \in L, \forall s \in R_L(x), t, z \in \{0, 1\}^*$,

$$|\Pr_U[\{D(S_{V^*}(x, t), z) = 1\}] - \Pr_U[\{D(View_v[(\tilde{P}(s) \leftrightarrow V^*(t))(x)], z) = 1\}]| \leq \epsilon(|x|).$$

This proof proceeds exactly as in Theorem 1, so we omit further details here. ∎

A.3 Proofs of Knowledge

Theorem 3: *The interactive proof system (P, V) is a proof of knowledge knowledge iff the system $\mathcal{P}^{pp} \times V$ is witness convincing for L.*

Proof. For the "if" direction of the proof, suppose that (P, V) is an interactive proof system for L that is a proof of knowledge. It is sufficient to show that the system $P^* \times V$ is witness convincing for L for every prover protocol P^*. By the definition of proofs of knowledge, for every prover protocol P^*, there is a negligible function ϵ_{P^*} and a probabilistic Turing machine E_{P^*} that takes as input the common input x and prover's auxiliary information s, runs in expected time polynomial in $|x|$, and outputs a candidate witness for x such that for all $x, s, t \in \{0,1\}^*$,

$$\Pr_U[\{Accepts_v[(P^*(s) \leftrightarrow V(t))(x)]\}] - \Pr_U[\{E_{P^*}(x,s)) \in R_L(x)\}] \le \epsilon_{P^*}(|x|).$$

Thus,

$$\Pr_U[\{Accepts_v[(P^*(s) \leftrightarrow V(t))(x)]\}] - \Pr_U[\{(s,t,x,E_{P^*}(x,s)) \in R_L^+\}] \le \epsilon_{P^*}(|x|), \qquad (1)$$

where the first probability $\Pr_U$ is taken over the random choices made by the prover and verifier protocols, while the second is taken over the random choices of E_{P^*}.

Define $\mathbf{M} : \mathcal{TM} \to \mathcal{EPPT}$ and $\epsilon : \mathcal{TM} \times \mathbb{N} \to [0,1]$ by taking $\mathbf{M}(P^*) = E_{P^*}$ and $\epsilon(P^*, \cdot) = \epsilon_{P^*}$. Suppose that there exists some $0 \le \lambda \le 1$ and some run r such that $(P^* \times \{V\}, r, 0) \models pr_p^{\lambda}(\Diamond \mathtt{accepts})$. Recall that $Pr_{(P^* \times V)(r)}$ can be identified with the uniform distribution $\Pr_{U_3}$ over triples of random strings. So $\Pr_U[\{Accepts_v[(P^*(r_p(0)) \leftrightarrow V(r_v(0)))(r_c(0))]\}] \ge \lambda$. By (1), we have

$$\Pr_U[\{(r_p(0), r_v(0), r_c(0), E_{P^*}(r_c(0), r_p(0))) \in R_L^+\}] \ge \lambda - \epsilon(P^*, |r_c(0)|).$$

Moreover, we have $(P^* \times V, r, 0) \models K_p(pr_p^{\lambda-\epsilon}(\mathtt{at\ time\ 0}\ \mathbf{M}_{p,R}))$, and so $(P^* \times V, r, 0) \models G_p^{\mathbf{M},0,\lambda-\epsilon} R$. This completes the "if" direction of the proof.

For the "only if" direction, let (P, V) be an interactive proof system for L such that the system $\mathcal{P}^{pp} \times V$ is witness convincing for L. Thus, there exist functions $\mathbf{M} : \mathcal{TM} \to \mathcal{EPPT}$ and $\epsilon : \mathcal{TM} \times \mathbb{N} \to [0,1]$ such that for every PPT $P^* \in \mathcal{P}^{pp}$, $\epsilon(P^*, \cdot)$ is a negligible function, and for every $0 \le \lambda \le 1$,

$$\mathcal{P}^{pp} \times V \models \mathtt{at\ time\ 0}\ pr_p^{\lambda}(\Diamond \mathtt{accepts}) \Rightarrow G_p^{\mathbf{M},0,\lambda-\epsilon} R_L^+. \qquad (2)$$

Let $E_{P^*} = \mathbf{M}(P^*)$ for every $P^* \in \mathcal{P}^{pp}$. Given a prover protocol P^*, define λ_{P^*} so that for all $x, s, t \in \{0,1\}^*$, $\Pr_U[\{Accepts_v[(P^*(s) \leftrightarrow V(t))(x)]\}] = \lambda_{P^*}(x,s,t)$. Since the probability distribution $Pr_{\mathcal{P}^{pp} \times V(r)}$ on $\mathcal{P}^{pp} \times \{V\}(r)$ can be identified with $\Pr_{U_3}$, it follows that

$$(\mathcal{P}^{pp} \times \{V\}, r, 0) \models pr_p^{\lambda}(\Diamond \mathtt{accepts})$$

for all $r \in P^* \times V$. By (2), for all $r \in P^* \times V$, we have

$$(\mathcal{P}^{pp} \times \{V\}, r, 0) \models G_p^{\mathbf{M},0,\lambda-\epsilon} R_L^+;$$

that is, $(\mathcal{P}^{pp} \times V, r, 0) \models K_p(pr_p^{\lambda-\epsilon}(\mathtt{at\ time\ 0}\ \mathbf{M}_{p,R_L^+}))$. We can view $(s,t,x,\mathbf{M}(P^*)(x,s)) \in R_L^+$ as a random variable (the probability that it is 1 is just the probability that $\mathbf{M}(P^*)(x,s)$ returns a y such that $(x, y) \in R_L$, taken over the random choices of $\mathbf{M}(P^*)$. Since $\mathbf{M}(P^*) = E_{P^*}$, we have that, for all $x, s, t \in \{0,1\}^*$,

$$\Pr_U[\{(s, t, x, \mathbf{M}(P^*)(x, s)) \in R_L^+\}] \geq \lambda_{P^*}(x, s, t) - \epsilon(P^*, |x|).$$

That is,

$$\Pr_U[\{Accepts_v[(P^*(s) \leftrightarrow V(t))(x)]\}] - \Pr_U[\{E_{P^*}(x, s)) \in R_L(x)\}] \leq \epsilon(P^*, |x|)$$

for all $x, s, t \in \{0,1\}^*$. It follows that (P, V) is a proof of knowledge. ∎

Acknowledgements

The first and third authors are supported in part by NSF grants ITR-0325453, IIS-0534064, and IIS-0812045, and by AFOSR grants FA9550-08-1-0438 and FA9550-05-1-0055. The second author is supported in part by NSF CAREER Award CCF-0746990, AFOSR Award FA9550-08-1-0197, BSF Grant 2006317 and I3P grant 2006CS-001-0000001-02.

References

Bellare, M. and O. Goldreich (1992). A modular approach to the design and analysis of authentication and key exchange protocols. In *Proc. CRYPTO '92*, pp. 390–420.

Dwork, C., M. Naor, and A. Sahai (1998). Concurrent zero-knowledge. In *Proc. 30th ACM Symposium on Theory of Computing*, pp. 409–418.

Fagin, R., J. Y. Halpern, Y. Moses, and M. Y. Vardi (1995). *Reasoning About Knowledge*. Cambridge, Mass.: MIT Press. A slightly revised paperback version was published in 2003.

Feige, U., A. Fiat, and A. Shamir (1987). Zero knowledge proofs of identity. In *Proc. 19th ACM Symposium on Theory of Computing*, pp. 210–217.

Feige, U. and A. Shamir (1990). Witness indistinguishability and witness hiding protocols. In *Proc. 31st IEEE Symposium on Foundations of Computer Science*, pp. 416–426.

Goldreich, O. (2001). *Foundations of Cryptography, Vol. 1*. Cambridge University Press.

Goldwasser, S., S. Micali, and C. Rackoff (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on Computing 18*(1), 186–208.

Halpern, J. Y., Y. Moses, and M. R. Tuttle (1988). A knowledge-based analysis of zero knowledge. In *Proc. 20th ACM Symposium on Theory of Computing*, pp. 132–147.

Halpern, J. Y. and M. R. Tuttle (1993). Knowledge, probability, and adversaries. *Journal of the ACM 40*(4), 917–962.

Moses, Y. (1988). Resource-bounded knowledge. In *Proc. Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pp. 261–276.

Rantala, V. (1982). Impossible worlds semantics and logical omniscience. *Acta Philosophica Fennica 35*, 18–24.

Tompa, M. and H. Woll (1987). Random self-reducibility and zero knowledge interactive proofs of possession of information. In *Proc. 28th IEEE Symposium on Foundations of Computer Science*.